2011

Combined Network and Erasure Coding Proactive Protection for Multiple Source Multicast Sessions

Suhas Shetty
Iowa State University

Combined network and erasure coding proactive protection for multiple source multicast sessions

by

Suhas Shetty

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:

Ahmed E Kamal, Major Professor

Aditya Ramamoorthy

Yong Guan

Iowa State University

Ames, Iowa

2011

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my Major Professor Dr. Ahmed E Kamal for his guidance and support throughout my research work. His encouragement and valuable inputs have been very instrumental in shaping this thesis work. I have enjoyed being a part of his research team and working with him has been a very gratifying experience.

I also want to extend my sincere thanks to my committee members Dr.Aditya Ramamoorthy and Dr.Yong Guan for their support.

I am greatly indebted to my parents Satish and Shashikala Shetty and my sister Shipali Shetty for their love and support throughout my graduate studies. I am also indebted to Santosh Kumar Chilkunda, Avinash Srinivasa and Poorvi Joebert without whom my master's journey would be incomplete. I would also like to thank all my friends in Ames for their support and making my stay here pleasurable.

I am thankful to Dr.Sudarshan Iyengar for motivating me and without whom I would not have pursued Graph Theory and Computer Networks as my graduate research interest.

ABSTRACT

Due to the increase in the density and multitude of online real time applications involving group communication which require a certain degree of resiliency, proactive protection of multicast sessions in a network has become prominent. The backbone network providing such a service hence needs to be robust and resilient to failures. In this thesis, we consider the problem of providing fault tolerant operation for such multicast networks that have multiple sources and need to deliver data to pre-defined set of destinations. The data is assumed to be delivered in a highly connected environment with a lot of nodes, e,g., a dense wireless network.

The advent of network coding has enabled us to look at novel ways of providing proactive protection. Our algorithm combines network and erasure coding to present a scheme which can tolerate predefined amount of failures in the paths from the sources to the destinations. This is accomplished by introducing redundancy in the data sent through the various paths. Network coding seeks to optimize the resource usage in the process. For sources that cannot meet the constraints of the scheme, protection is provided at the cost of reduced throughput.

CHAPTER 1. INTRODUCTION

1.1 Overview

Several applications like Teleconferencing, video, audio on –demand services consume a lot of network bandwidth. In a uni-casting network environment where several users are requesting the same video or if they are participating in the same teleconference, a number of copies of data packets are required to be sent, leading to large bandwidth consumption. It has been observed that for such applications, multicasting is a more useful and effective operation. Using multicast services, data packets can be sent from one source to several destinations by sharing the link bandwidth. An important aspect of providing reliable service through multicast is by ensuring that the network is fault tolerant and users still receive data in spite of path failures.

Many algorithms have been deployed to employ recovery methods for multicast networks. They are broadly classified into on demand and pre-planned approaches. On- demand approaches (Reactive protection) do not compute backup paths beforehand, but do it after a failure in the network has occurred. Hence they have a longer recovery time. This is not desirable and acceptable in several real time applications which require non-interrupted communication. The second approach toward failure recovery is called a pre-planned approach (Proactive protection). As the name indicates the backup routes in this case have been pre-defined. Though the resources required for this approach is more, it provides for a very short recovery time.

Proactive protection can be achieved in many ways. Some of the primitive techniques consume almost twice the bandwidth required by the application. These

involved providing backups for links and paths separately, much of which was never utilized when there were no failures. Hence a considerable amount of bandwidth went unutilized.

With the advent of network coding [1], concepts from information theory were widely initiated into solving research problems in networks. One such idea is borrowed from the Forward Error Correction method called erasure coding used in binary erasure channels. This involves transforming a smaller message of k symbols into a larger message of say n symbols such that a subset of k symbols is enough to recover the original message. The same method has also been employed in cryptography for key sharing and is called Shamir's Threshold Sharing Scheme (TSS) .Authors in [2] applied this scheme for fault tolerance in multicast networks. In the scheme, n disjoint trees from the source to the multicast destinations carry packets that contain the original as well as redundant data. At any of the receiver only k out of the n packets are sufficient for the recovery of original data. Hence the scheme can support breakdown of up to n-k paths. We extend this idea for multicast networks involving multiple source inputs. Using network coding, we reduce the amount of overhead needed to deliver data to the receivers from multiple sources while retaining network resilience.

1.2 The Multicast Fault Tolerance Problem and its Applications

Multicast involves sending data from one source to various destinations. The network through which the data is being sent has to be resilient to breakdowns. The problem is compounded when there are multiple sources which need to transmit data. We deal with the problem in which the network has to be robust against edge failures.

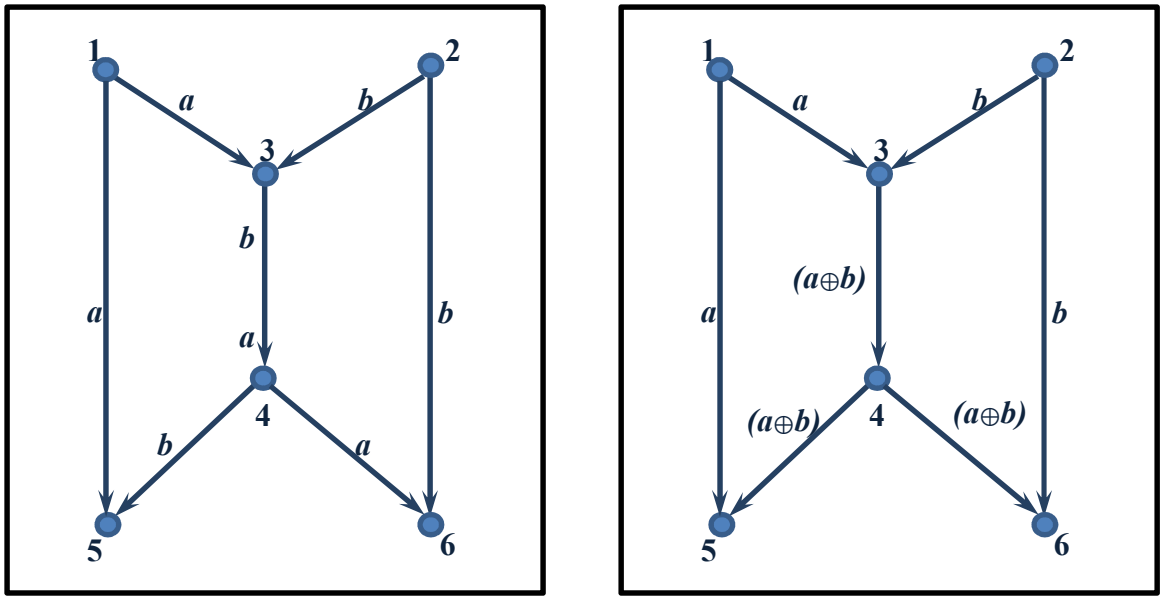
Consider a scenario where there are multiple camera feeds which need to be transmitted to a subset of the audience in a room, where the member belonging to the subset need to receive all video feeds. Here there are a small number of sources and a large number of destinations and nodes. The nodal degree of the network graph of the audience would be reasonably large, enabling us to find lot of disjoint paths to any given person in the audience. The data carried across some of these paths will be the actual data whereas across some will be redundant data introduced into the network to achieve fault tolerance.

This introduction of data is possible through erasure coding which was discussed before. We encode the original data and send parts of it through n paths such that data from k paths is enough to recover the original data. This allows us the liberty of being able to support the breakdown of n-k paths. Since feeds are coming in from multiple sources we need to reduce the amount of overhead or redundant information being sent on the network to be able to utilize the bandwidth effectively. Through the use of network coding we combine the data of all the sources which are being sent on those n-k paths to save on the data overhead.

1.3 Network Coding

Network coding is a technique for efficient utilization of bandwidth. The idea here is to combine instead of repeat. In any give multicast network with multi source input, there will be several links which will be carrying data from different sources to multiple destination nodes. In normal transmission if a router has to send say two such multicast packets over a link it would take two time units. With network coding

it is possible to send both data units in a single unit of time. This is demonstrated via a simple butterfly network shown below.



(a) (b)

Figure 1.1 Two Source, Two destination Graph a)Without Network Coding b)With Network Coding

Consider a scheme with two sources **1** and **2** and two destinations **5** and **6**.Let a be the data packet of Source **1** and b be the data packet of Source **2**.Each of the destinations needs data from either of the sources. The network graph is shown above. In Figure 1.1 (a), the node **3** needs two time units to send a and b to node **4** since there is a single link between **3** and **4**. But as we see in Figure 1.1 (b), if we XOR a and b and send it over the link connecting node **3** and node **4,** we will still be able to recover **b** at node **5** and a at node **6**. This is possible because, if we XOR a which node **5** receives from node **1** with $a \oplus b$, *we* obtain b *(*similarly we can obtain a at node **6**). In practical network coding, instead of XORing which is addition in the field F_2, we use operations over a field of higher order.

1.4 Existing Protection Schemes

There have been several approaches for providing fault tolerance to multicast networks. Some of the primitive techniques of protection involved providing backup path to every link or possible primary path in the network. In the Dual-tree protection scheme [3], instead of protecting each link or member individually in the multicast tree, a secondary tree is built amongst a subset of the multicast members. This scheme requires the underlying topology to be a bi-connected graph, with the constraint that there are at least two vertex disjoint or link disjoint paths between any two nodes. It can also be noted that unlike the link and path protection approaches, dual-tree scheme does not require per-link or per-path fault-tolerance management.

Aggregated MPLS-based Fault Tolerant Multicast [4] is based on the concept of aggregated multicast. The idea here is to have one multicast tree support more than one multicast groups, making it a many-one mapped tree. The advantage is that routers will have to maintain states of fewer multicast trees. It is easier for them to lookup for backup paths and the number of table entries would be fewer.

The Modified Threshold Sharing Scheme [2] provides protection through data redundancy. Data shares are sent on n disjoint paths and only k of them are enough to recover data at the receiver. It employs Shamir's key sharing algorithm (or the concept of erasure coding) to generate such shares. This method is the basis for the proposed approach. It is discussed in detail in the next session.

Network coding is being widely used for effectively utilizing bandwidth. It is also increasingly being employed to provide protection against faults. With prior knowledge of edge failure patterns, network codes can be constructed guaranteeing a certain

throughput in spite of breakdowns. The authors in [5] established the algebraic framework for linear network coding and also discussed recovery based on knowing edge failure patterns beforehand. Polynomial time construction of network codes that could handle such failures was discussed in [6]. In [7], an information theoretic framework to discuss network management in the presence of edge failures was presented. Authors in [8] employed network coding to protect against node failures.

While network coding for a single source multicast session has been researched a lot, results on multi source multicast sessions have not been that forthcoming. Pollution, where certain sinks receive data that is redundant from a decoding perspective has been a primary concern. The author in [9] proposed a constructive scheme that provided achievability theorems for multi-source multicast sessions. Authors in [10] partition the graph into sub graphs and transform the multi-source problem into a combinatorial optimization problem.

1.5 Erasure Coding Based Protection

1.5.1 Scheme

This approach [2] for fault tolerance is based on Adi Shamir's Threshold Sharing Scheme (TSS) in security. The scheme uses a modified version of TSS. The idea here is to divide the original message into say n different shares and route these on n disjoint multicast trees. These n shares are such that only k of them are required to recover the original message. Hence even if there are (n-k) failures the message can still be recovered at the destination. The scheme is called a (k,n) threshold sharing scheme. For example if the threshold scheme which is being used is **(3,5),** then out of the **5**(n) packets which leave the node, the receiver needs to receive only **3**(k) of them

to be able to recover the message. The recovery happens through the Lagrange interpolation method. The scheme above can support the loss of at most two **(2)** of the packets owing to link failures in two multicast trees. Since our proposed approach works on the same principle and is an extension of this scheme, understanding this scheme an example would be helpful to illustrate the concepts.

1.5.2 Modified TSS Example

The scheme which will be used in this example is a **(3,5)** threshold sharing scheme., i.e.,k **=3,** n**=5,**where

k - Number of paths which have to be intact in order to support message recovery

n - Total number of paths.

Let us divide the original Data packet into m blocks or shares ($\mathbf{S_0}$ **,**$\mathbf{S_1}$**....** $\mathbf{S_m}$), where

m- (Total packet size in bytes/L) and

L - size of each block and is defined as

$L = h^*k$, where h can be any integer,

We have chosen h =1.Hence L=3 and the size of the block becomes 3.

1.5.3 Message Distribution Process

In the message distribution process the generator functions encode the original set of **3** packets into a total of **5** packets as shown. This is then transmitted on the network. If any of the receivers receive fewer than **5** packets (greater than **3** packets) due to failure of edges in the network, the original packet will still be decodable.

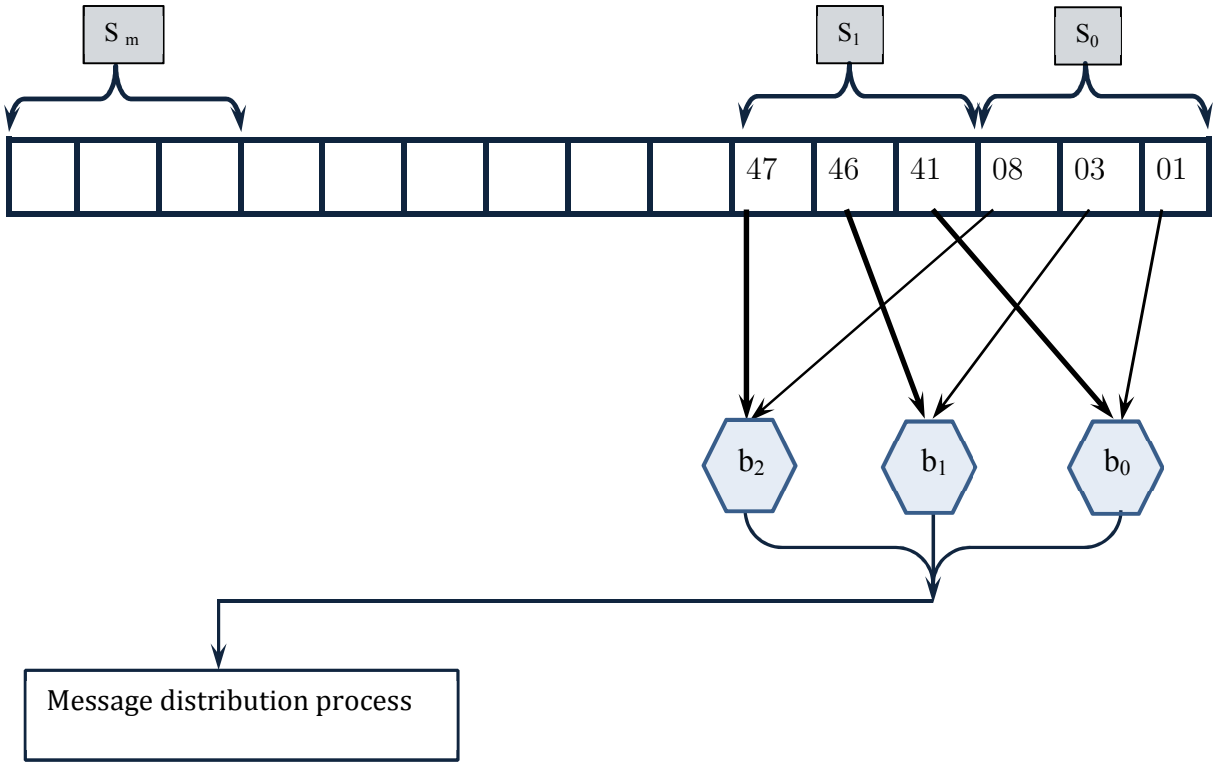


Figure 1.2 Distributing the Original Data Packet

If p is a prime number a polynomial of degree k-1 in x over Z_p is defined as

$$f(x) = (b_{k-1}x^{k-1}+\ldots\ldots b_2x^2+b_1x+b_0) \bmod p$$

The b_i's are chosen from the original data packet as shown in Figure 1.2. The message distributor generates $f(x)$ for values of x over the field Z_p. Figure 1.2 shows the selection of b_i's and Figure 1.3 shows the generator functions. The various $f(x)$ values are then sent on the disjoint trees.

The decoding at the receiving end is accomplished using the Lagrange interpolation method.

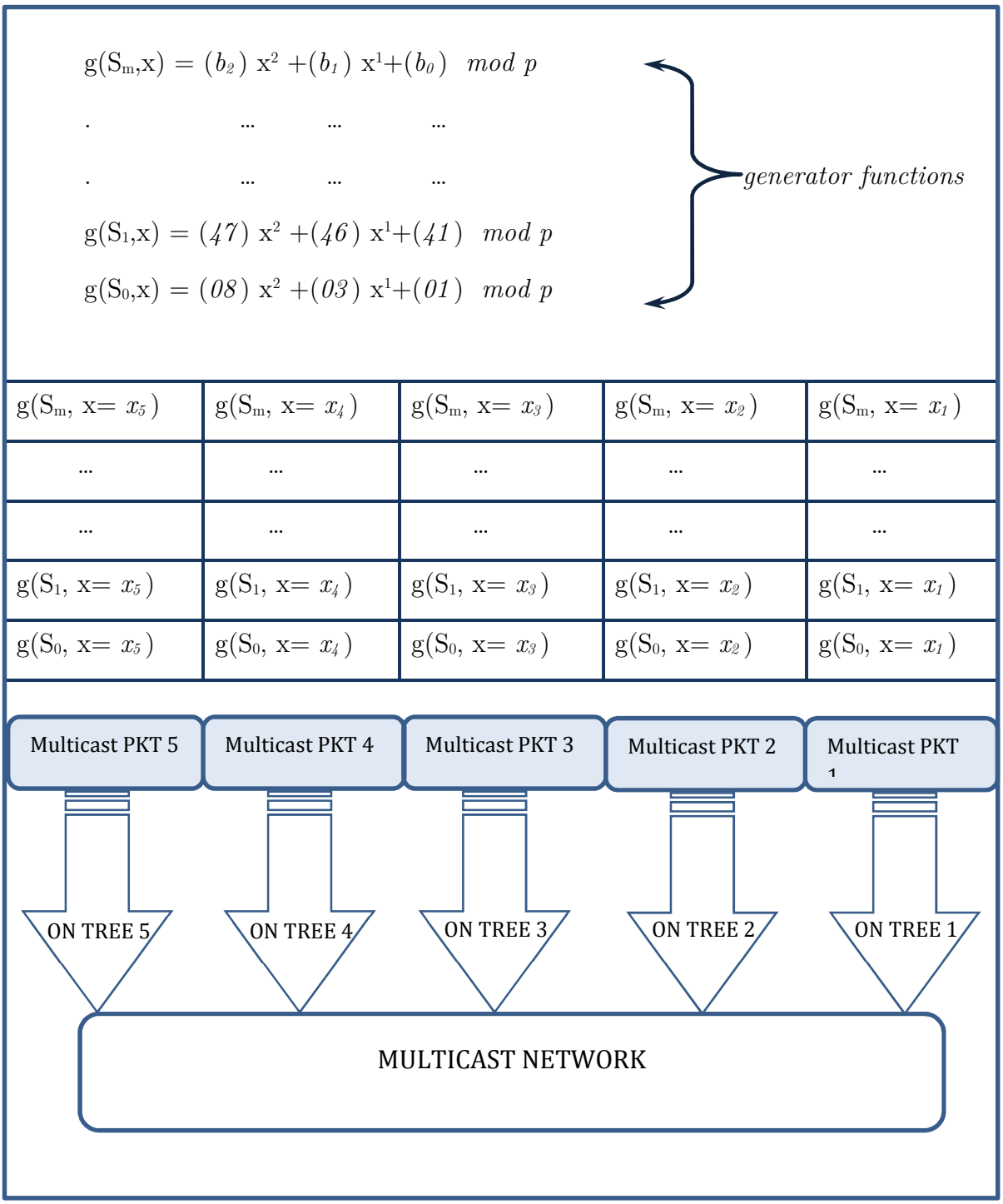


Figure 1.3 Message Distribution Process

1.5.4 Advantages and Disadvantages

This scheme has several advantages. It can support multiple link failures very easily. The algorithm can support different levels of resilience based on users' requirements. Fault tolerance is present by default and there is no router state update required in order to recover from link failures. Hence there is almost zero delay in recovering from link failures. There is no backup tree computations as they can be combined as part of the n disjoint multicast trees required to route the n packets.

The disadvantages of this method are that the algorithm imposes greater connectivity requirement on the network owing to the n disjoint trees which are required to route the multicast packets. Also, additional bandwidth is required since the amount of information sent per packet is greater than the original size of the packet, and is a factor of the number of link failures which need to be supported. This bandwidth will increase when there are multiple sources that need to be supported. Our algorithm tries to address this problem by provisioning trees belonging to multiple sessions on the same tree and using network coding to encode the data.

1.6 Thesis Contributions

The work in this thesis tries to address the problem of reducing the resources utilized by the erasure coding method. We consider a network where the erasure coding scheme is employed by many sources. Every source requires at least n disjoint paths to all the destinations. These paths to multiple destinations form a multicast tree. Erasure coding alone would have necessitated all n of these paths of every source to be disjoint. Using network coding we are able to remove this constraint. There would still be k disjoint paths, but n-k (the number of paths the breakdown of which

can be supported) paths used for protection can be combined so as to reduce the overall overhead. The thesis discusses the problem and the set of flow constraints that need to be satisfied by the participating sources. The algorithm also tries to provide the required resilience at the cost of decreased throughput if certain sources are not able to meet the demands of path disjointedness and the flow requirements. We show that our scheme outperforms the erasure method based protection in terms of resource utilization while providing almost the same protection.

1.7 Thesis Organization

The thesis is organized as follows: In the Problem Formulation section we discuss the fault tolerance problem, the flow constraints that need to be satisfied by the various participating nodes in the network. The proofs for the sufficiency of these conditions are also discussed. The next section discusses the algorithms to find out if the nodes meet the required constraints. If they do not meet the same, a scheme to selectively discard nodes based on their ability to compromise throughput for resilience is formulated. This is followed by the algorithm to check the condition for shared trees. On meeting all these constraints actual multicast trees are established which determine the cost savings.

The coding coefficients for the network coding scheme are discussed next. This involves the field size discussion and the scheme itself based on Vandermonde matrix .The sufficiency of the scheme for receiver decoding is also established.

The simulation results demonstrate the performance of our method against erasure coding based scheme. It shows the savings in terms of cost of provisioning this protection. A comparison of both the schemes for various values of k and n is also

demonstrated via graphs. The thesis concludes with further improvement and future research work that can be performed.

CHAPTER 2. PROBLEM FORMULATION

2.1 Introduction

This section explains the problem and its formulation in detail. It specifically discusses the various constraints which need to be satisfied by the nodes in order for them to be a part of the algorithm. It also proves that these constraints are necessary and sufficient in order to ensure that the algorithm will work.

2.2 Multiple Source Multicast Session Protection

2.2.1 Network Model and Symbols

This section details the network graph and how it is modeled and symbols for different problem parameters.

Symbol	*Definition*
N	Total number of nodes in the graph
s	Total number of sources
r	Total number of sinks
S_i	The i^{th} source node
S	Set of all the sources
t	Used to denote a sink
T	Collection of all the sinks
T_j	The j^{th} source node
n	The total number of shares
k	The number of shares required to decode the data
q	Total number of shared graphs (n-k)
$f_{in}(x)$	The flow incoming to node x
$f_{out}(x)$	The flow outgoing from node x
$f_{max}(x,y)$	The maximum flow from node x to node y
$f^{r}_{m}(S_i,T)$	Multicast flow(rate) from S_i to each of the destinations in T.

Table 2.1 Network Symbols 1

The network is modeled as an undirected graph *G (V, E)* with *V* as the set of nodes and *E* as the set of undirected edges. The edges are assumed to be of unit capacity. All the sources transmit data to every destination. The various symbols which are used is this section are mentioned above in the table

The erasure coding scheme which would be used will be represented as *(k,n).*This tolerates breakdown of *n-k* paths without of loss of data. For sources that cannot satisfy the requirement of *n* paths (*k* disjoint and *n-k* shared paths), there are several approaches that can be taken.

1) Throughput only compromise: This approach involves compromising the throughput (*k*) in order to provide the guaranteed protection (*n-k*). Every source will specify a k^{th} which is the minimum threshold of throughput required by the particular source. We reduce *k* till we reach the threshold. (*n* also reduces keeping *n-k* constant). This approach is generally used when the source cannot meet the requirement of *k* disjoint paths.
2) Protection only compromise: This approach involves compromising protection (*n-k*) without reducing the throughput (*k*). This can be accomplished by reducing *n* (and keeping *k* constant) which decreases the number of shared trees (*n-k*) of which the source is a part.
3) Throughput and Protection compromise: A combination of both approaches mentioned before is possible. We first reduce the throughput (*k*) as long as it remains above the k^{th} (*n* also reduces keeping *n-k* constant). If this does not yield us enough disjoint and shared paths then we reduce *n* (keeping *k* constant) to reduce protection (*n-k*). A reduced protection is provided if the source cannot meet the constraints in spite of reducing throughput.

2.2.2 Problem Definition and Solution Approach

The multiple source multicast fault tolerance problem can be formally defined as follows:

Problem Definition: Given the network *G (V, E)* consisting of *N* nodes with *s* sources and *r* receivers, every receiver requiring data from every source, provision a set of *n* multicast session connection paths between every source and all the receivers such that

a) Every receiver receives at least *k* shares on *k* disjoint paths,

b) The breakdown occurs for at most *n-k* paths between any source and receiver at any point of time

c) Minimum cost is required for provisioning sessions

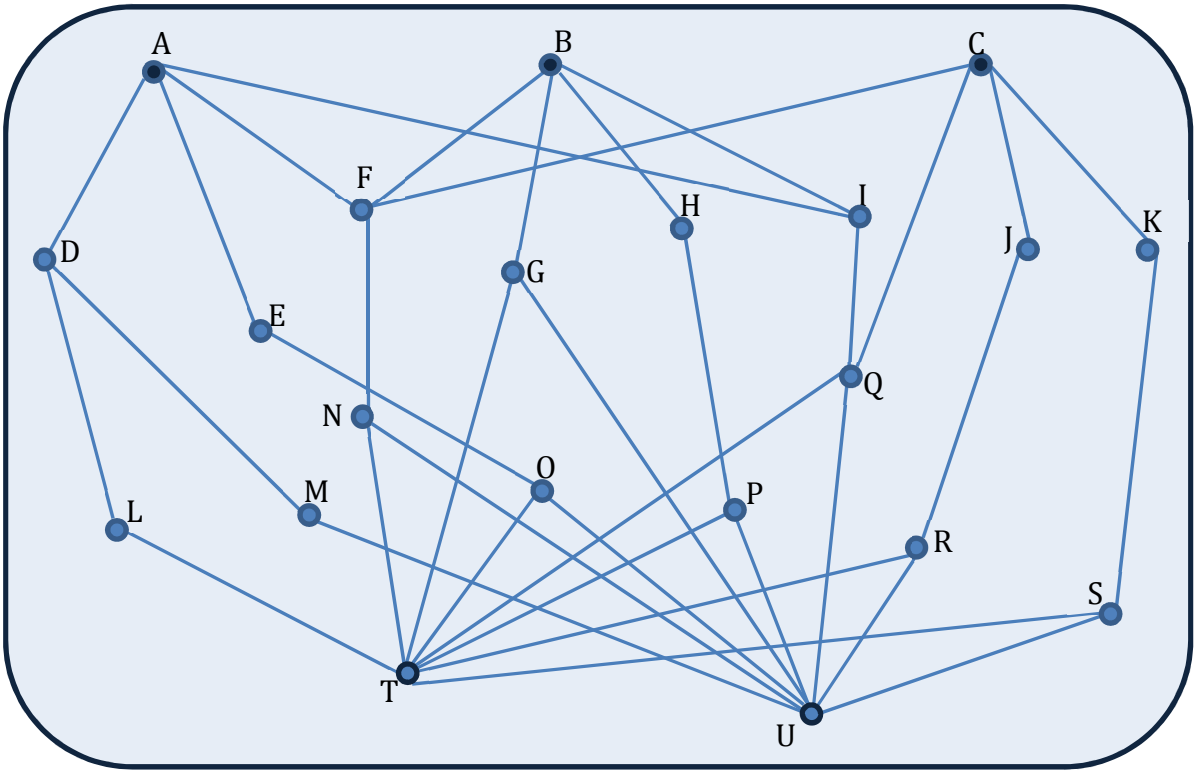
d) Maximum number of sources are supported in the event of network being unable to support all, at reduced throughput (exceeding k^{th}) and/if necessary protection.

The solution approach discussed in this thesis can be briefly described as follows

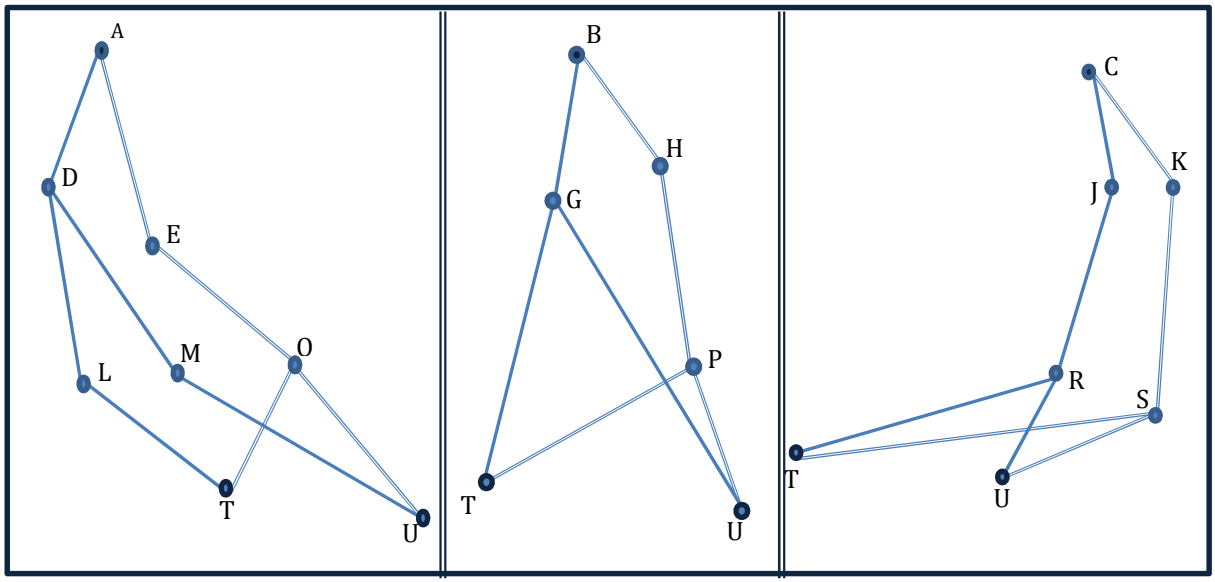
1) The first step in the scheme is checking for all the sources and destinations to see if they meet the necessary and sufficient flow constraints.
2) If sources meet all the conditions then actual multicast trees (both disjoint and shared) are calculated. A graph example has been discussed below for better understanding.
3) If they do not, then the throughput and protection compromise approach is followed. Sources that do not meet these constraints will be discarded.
4) The generator functions along with the prime number and field size for them is decided amongst the sources and the receivers.
5) The coefficients for the network coding are agreed to between the sources and receivers and encoded in the metadata of the packets.

2.2.3 Multicast Protection Example

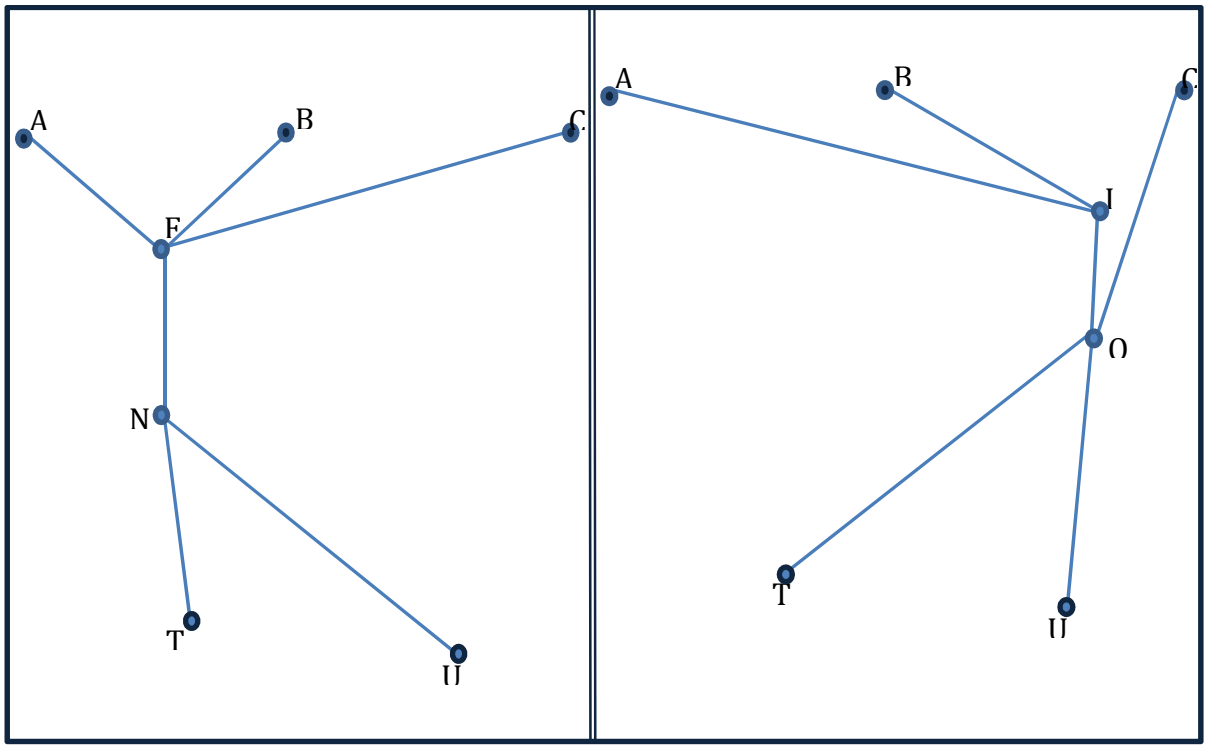
An example network is shown in Figure 2.1 (a) and consists of 21(N) nodes. There are three sources (s=**3**) **A, B** and **C** and two destinations (r=**2**) **T, U**. n is 4 and k is 2. Hence two disjoint trees for each of the sources are shown in Figure 2.2 (b).The two shared trees are shown in Figure 2.3 (c). For the shared tree represented by the following edges {**A-F, B-F, C-F, N-F, N-T, N-U**}, the erasure coded packets from the sources **A, B** and **C** will be network coded at the node F. For the shared tree {**A-I, B-I, I-Q, C-Q, Q-T, Q-U**}, erasure coded packets from **A** and **B** are combined at node **I** which is then combined with the packet from node **C** at node **Q**.



(a)



(b)



(c)

Figure 2.1 a) Network graph with N=21 b) Disjoint trees for sources **A, B** and **C** respectively c) Shared trees for nodes **A**, **B** and **C**

2.3 Flow Constraints

2.3.1 Theorems and Proofs for Necessary Conditions

The below theorems and their subsequent proofs are necessary and need to be satisfied by the nodes to be able to be a part of the scheme.

Theorem 1. $f_{in}(t) \geq \quad k * s + (n\text{-}k) \; \forall t \in T$

Proof. We need k disjoint trees between every source and destination in addition to n-k trees which can be shared with other sources. Since an incoming edge to a destination from any tree corresponds to a flow of one (every edge capacity is one), therefore every destination will need an incoming flow of at least $k*s$ for the s sources (Since the trees from all the sources have to be disjoint simultaneously). Additionally there is a shared flow of n-k which will come from all the sources put together. So the total incoming flow to the any destination should at least be $k * s + (n\text{-}k)$.

Theorem 2. $f_{out}(S_i) \geq \quad n \; \forall S_i \in S$

Proof. Since for any given source n packets have to be sent on n different trees, we need an outgoing flow of at least n from each source (since one outgoing tree corresponds to an outgoing flow of one).

Theorem 3. $f_{max}(S_i, \; t) \geq \; n, \forall S_i \in S$ and $t \in T$

Proof: Similar to the above reasoning since there needs to be n packets sent from any source to any destination on n disjoint paths, and since edge capacity is one, a minimum flow of n is required between any source and any destination.

2.3.2 Theorems and Proofs for Necessary and Sufficient Conditions

These conditions are necessary and sufficient for the required disjointedness and flow in the graph. The first set of conditions is for the disjoint multicast trees. The second set of conditions is for the trees shared across different multicast sessions.

The multicast flow through one multicast tree is defined as sum of the flows received by all the destinations .So if there are r destinations in the multicast tree, the multicast flow(assuming a flow of one is received by each destination from the source) is defined as r.

All receivers need to receive equal flow.

$f_m(S_i, T)$ – Multicast flow(rate) from S_i to each of the destinations in T.

a) Disjoint trees

Theorem 4.

$f_m(S_1, T) \geq r * k$

$f_m(S_2, T) \geq r * k$

.

.

$f_m(S_s, T) \geq r * k$

Proof: Theorem 4 is for the case of disjoint trees. According to the problem statement we need at least k disjoint multicast trees from any source to all the destinations So a k disjoint tree requirement from source to all the destinations implies a minimum multicast flow of $r*k$.. (There will be k disjoint trees from the source to the r destinations). Since the k disjoint trees of any given source are distinct from the k disjoint trees of any other source, all the conditions of **Theorem 4** have to hold true

simultaneously. Hence there has to be a multicast flow of at least $r*k$ from every source to all its destinations.

Alternatively, if all the conditions of **Theorem 4** are true, it implies that there is a flow of $r*k$ from every source to its multicast destination at the same time. Because of the unit capacity edge condition, these $r*k$ flows for every source are disjoint from the others. Since a multicast flow by definition means that every destination has to receive an equal amount of flow, the previous statement implies that there is a flow of k from any given source to any destination, i.e k disjoint paths. In essence, there are k disjoint trees between every source – every destination.

b) Shared graphs

Theorem 5.

$\beta_1 f'_m(S_1,T) + \beta_2 f'_m(S_2,T) + \ldots\ldots + \beta_s f'_m(S_s,T) \geq (n-k)*r$

where $(\beta_1, \beta_2 \ldots\ldots \beta_s)$ (s-tuple) is set of all binary vectors : $(\beta_1, \beta_2 \ldots\ldots \beta_s)_{10} = 2^y, 0 \leq y \leq s-1$

Proof: Theorem 5 is for the case of shared trees. The goal is to ensure that every destination gets at least $n-k$ copies from every source. These copies are network coded - The data from various sources are combined and sent. These $n-k$ copies will result in $n-k$ equations with s unknowns (for s sources) in the extreme case where all source data is combined into one single copy for every shared tree.

To ensure that $n-k$ multicast copies of every source reach every destination, we need at least $n-k$ disjoint ways of reaching every destination. i.e $n-k$ multicast trees between any source and the corresponding destinations. In **Theorem 5** the condition has to hold for any value of the binary vector. That translates to every source needing to have a multicast flow of at least $(n-k)*r$ to the various destinations. Note that

Theorem 4 and **5** have to be valid simultaneously in the network. An alternate way of stating is that **Theorem 5** has to hold in the graph obtained by removing the edges contributing to the validity of **Theorem 4.**

Here trees across sources can have common edges because on those edges we can send data in a network coded fashion. So essentially, the network graph left after removing the edges which are a part of the flows in **Theorem 4** should have at least *n-k* disjoint paths from any source to any destination. When considering all the sources simultaneously edges which are common can send network coded data, whereas edges that are exclusive to trees can carry the original data as is.

If **Theorem 5** is satisfied for every possible binary vector it implies that there is a flow of at least *r*(n-k)* from every source to all the destinations in the network obtained by removing edges involved in supplying the required flow in **Theorem 4**. Since this is a multicast flow for r destinations it implies that every destination receives a flow of *n-k*. That is there are *n-k* different paths to reach a destination from any given source. Hence *n-k* different network coded copies of the original data can be obtained by any destination.

2.3.3 Flow Constraint Example

The theorems stated above can be explained using the network graph in Figure 2.1 (a). The total number of nodes $N = 21$. The number of sources is 3 (s=3). There are two destinations (r=2). The requirement for the total paths and total disjoint paths is 4 (n) and 2 (k) respectively.

The source nodes are **A, B** and **C** and the destinations are **T** and **U**.

1) Theorem 1: $f_{in}(t) \geq \quad k * s + (n-k) \; \forall t \in T$

The incoming flows to node **T** and **U** are 8 each,

L.H.S: $f_{in}(\mathbf{T}) = 8$, $f_{in}(\mathbf{U}) = 8$,

R.H.S: $k*s+ n-k = 8$,

Hence Theorem 1 holds good

2) Theorem 2: $f_{out}(S_i) \geq \quad n \; \forall S_i \in S$

The outgoing flows from nodes **A, B** and **C** are 4 each.

L.H.S $f_{out}(\mathbf{A}) = 4$, $f_{out}(\mathbf{B}) = 4$, $f_{out}(\mathbf{C}) = 4$,

R.H.S: $n=4$

Hence Theorem 2 holds good

3) Theorem 3: $f_{max}(S_i, t) \geq n, \forall S_i \in S$ and $t \in T$

From the Figure 2.1 (a), the max flows from **A, B** and **C** to **T** and **U** are equal to 4,

L.H.S: $f_{max}(\mathbf{A}, \mathbf{T}) = 4$, $f_{max}(\mathbf{A}, \mathbf{U}) = 4$, $f_{max}(\mathbf{B}, \mathbf{T}) = 4$, $f_{max}(\mathbf{B}, \mathbf{U}) = 4$,

$f_{max}(\mathbf{C}, \mathbf{T}) = 4$, $f_{max}(\mathbf{C}, \mathbf{U}) = 4$

R.H.S:$n = 4$

Hence Theorem 3 is valid

4) Theorem 4:

$f_m(S_1, T) \geq \quad r * k$

$f_m(S_2, T) \geq \quad r * k$

.

.

$f^{r}_{m}(S_s, T) \geq \quad r^* k$

L.H.S: $r^*k = 4$,

Hence we need a multicast flow of at least 4 on the L.H.S.

In the network of Figure 2.1 (a) nodes **A, B** and **C** each have a multicast flow of 8. Since the Theorem 4 implies that every source needs to have a multicast flow of 4 that is edge disjoint from the others we consider Figure 2.1 (b). It shows two trees ({**A-D, D-L, D-M, M-U, L-T**} and {**A-E, E-O, O-T, T-U**}) for node **A** ,two trees for **B** ({**B-G, G-T, G-U**} and {**B-H, H-P, P-T, P-U**}) and two trees for **C** ({**C-J, J-R, R-T, R-U**} and {**C-K, K-S, S-T, S-U**}) which are edge disjoint from each other and contribute to a multicast flow of 4 for each node.

Since every tree of node **A, B** and **C** is disjoint from each other, all the equations in Theorem 4 are valid.

5) Theorem 5:

$\beta_1 f^{r}_{m}(S_1, T) + \beta_2 f^{r}_{m}(S_2, T) + \ldots\ldots + \beta_s f^{r}_{m}(S_s, T) \geq (n-k)^* r$

where $(\beta_1, \beta_2 \ldots\ldots \beta_s)$ (s-tuple) is set of all binary vectors : $(\beta_1, \beta_2 \ldots\ldots \beta_s)_{10} = 2^y, 0 \leq y \leq s-1$

R.H.S: $(n-k)^* r = 4$

We have $T = \{\mathbf{T}, \mathbf{U}\}$ and $\beta_1 f^{r}_{m}(\mathbf{A}, T) + \beta_2 f^{r}_{m}(\mathbf{B}, T) + \beta_3 f^{r}_{m}(\mathbf{C}, T)$ on the L.H.S.

$s = 3$, Hence the possible binary vectors such that $(\beta_1, \beta_2, \beta_3)_{10} = 2^y$ are $(0,0,1)$, $(0,1,0)$ and $(1,0,0)$.The three equations which need to be satisfied therefore are

$f^{r}_{m}(\mathbf{A}, T) \geq 4$

$f^{r}_{m}(\mathbf{B}, T) \geq 4$

$f^{r}_{m}(\mathbf{C}, T) \geq 4$

Note that at a time the binary vector can take only one value and hence the three flows need not be disjoint from each other. They still need to be disjoint from the flows used in Theorem 4. From Figure 2.1 (c), we have 2 multicast trees ({**A-F, B-F, C-F, F-N, N-T, N-U**} and {**A-I, B-I, I-O, C-O, O-T, O-U**}). For node **A** the first tree provides a multicast flow of 2 through {**A-F, F-N, N-T, N-U**} and the second tree provides a multicast flow of 2 through {**A-I, I-O, O-T, O-U**}**.** Hence **A** has a multicast flow of 4. Similarly it can be seen that nodes **B** and **C** too have a multicast flow of 4 from the two multicast trees. The edges of these trees being disjoint from the trees in Figure 2.1 (b), Theorem 4 and Theorem 5 hold simultaneously.

In the above example the shared trees in Figure 2.1 (c) network coding would happen on nodes **F**, **I** and **O**.

.

2.4 Summary

In this chapter we discussed the network graph and the problem formulation. Some of the details of the scheme mentioned will be discussed in the next section. We also saw a graph example and how the disjoint and shared paths were formed.

CHAPTER 3. MULTICAST TREE PROTECTION ALGORITHM

3.1 Introduction

In this chapter we discuss the algorithm for verifying the flow constraints discussed before. This includes the selection of destination nodes and source nodes that can take part in the scheme. This is followed by the algorithm to build actual multicast trees for the disjoint and shared trees.

3.2 Network Symbols and Algorithm Overview

The below table lists and defines all the symbols and terms used in this chapter. Symbols already defined in earlier chapters will also be used.

Symbol	*Definition*
S_i	The i^{th} source node
S`	Collection of all the sources
t_j	Used to denote a sink
T`	Collection of all the sinks
U_x	Set containing sources with a protection of n-k-x
$N^{gr}{}_x$	Total number of paths for the sources in the x^{th} group
$K^{gr}{}_x$	Total number of disjoint paths for the sources in the x^{th} group
g	Total number of groups
U_{ji}	The i^{th} source of the j^{th} group in U
f_{temp}	Multicast flow variable
$K^{th}{}_i$	Threshold value of number of disjoint paths(or throughput) for a source i
n_i	Number of total paths for a source i
k_i	Number of disjoint paths for a source i
f_{ij}	Flow path between source i and destination j
$f^k{}_{ij}$	Denotes the k^{th} flow path between source i and destination j
$m^i{}_j$	Max flows to destination j from source i
$F^{k_1\, k_2 \ldots\ldots\, k_r}$	Combination of various flow paths

$O(F)$	Ordered arrangement of flow combinations
M^F	Multicast flow value
$P(F^{k_1 k_2 \ldots k_r})$	No of paths (f^k_{ij}) with which $F^{k_1 k_2 \ldots k_r}$ shares common edges.
$C(F^{k_1 k_2 \ldots k_r})$	Measures percentage of throughput savings for flow combination $F^{k_1 k_2 \ldots k_r}$
F_{ji}	The i^{th} source of the j^{th} group in F
F_x	Contains sources in the x^{th} group
k_i^{old}	Number of disjoint paths for a source i before running algo 3.3.2c)
f_b^{tr}	Used to represent a tree formed by a collection of flows

Table 3.1 Network Symbols 2

Algorithm Overview

The flow verification step consists of algorithms that check whether Theorem 1-5 hold in the network or not.

1) Destination Selection: Checks whether the destination has enough incoming flow to handle connections from all the sources (Theorem 1).

2) Source Selection 1: This checks whether the source has enough outgoing flow to the destinations in order to support n paths. (Theorems 2 and 3)

3) Source Selection 2 and Grouping: Checks if the sources can satisfy theorems 4 and 5 i.e., whether they have enough disjoint and shared trees to support the proposed approach. Grouping is the process of segregating sources based on the amount of compromise in throughput and protection they require. (Theorems 4 and 5).

3.3 Destination Selection

The destination nodes that can satisfy the constraints discussed in Theorem 1 are selected in this algorithm. Since the destinations should be able to support k disjoint paths coming each of the sources (s), they need a minimum input flow of $k*s$ to

accommodate the same. The shared trees would contribute in the worst case *n-k* incoming edges at least. Essentially the destination needs to have a minimum of $k * s + (n-k)$ as the incoming flow. The destination selection step checks if the destined nodes can support this flow, and retains or discards destination nodes based on this. The algorithm is described below.

Algorithm 3.1 (A-3.1)

Algorithm	**Destination Selection** This algorithm filters out the destination nodes that cannot meet the inflow constraints.
Input	Network Graph G Destination Set T
Output	New Destination Set $T`$

```
1  T ` = ∅
2  foreach j← 1 to r do
3      if ( f_in(T_j) ≥  k *s + (n-k)) , T_j∈T
4          T `← T `∪ t_j
5      endif
6  endfor
```

3.4 Source Selection and Grouping

The sources in the network need to satisfy Theorems 2, 3, 4 and 5 to be able to participate in the propose approach. Specifically we need to verify whether the sources have enough flow to the destinations and enough outgoing edges to support the flow. The process of verification is divided into two steps. Source Selection 1 checks for the outgoing flow condition and max flow condition mentioned in Theorems 2 and 3.

Source Selection 2 and Grouping verify whether there are enough multicast flow to satisfy conditions in Theorems 4 and 5.

.

3.4.1 Source Selection 1

The first step, Source Selection 1, determines if the sources have the minimum number of outgoing edges and the minimum flow required to take part in the scheme .Since there are n paths originating from every source, the minimum outflow required is n. Also since there will be at least n multicast paths from any source to any destination, there has to be a minimum flow of n between any source and destination. If either of these conditions are not met, the source will be discarded.

Algorithm 3.2 (A-3.2)

Algorithm	**Source Selection 1** This algorithm filters out the source nodes that cannot meet the outflow constraints as well those that do not have substantial flow to the destination nodes.
Input	Network Graph G Destination Set T Source Set S
Output	New Source Set $S`$

1 $S` = \emptyset$
2 **foreach** $i \leftarrow 1$ *to* s **do**
3 **if** ($f_{out}(S_i) \geq n$), $S_i \in S$
4 **if** ($f_{max}(S_i, t) \geq n$, $\forall t \in T$ `)
5 $S` \leftarrow S` \cup S_i$
6 **endif**

7 **endif**
8 **endfor**

3.4.2 Source Selection 2 and Grouping

This stage involves second level of source filtering along with their segregation into groups. The order of sources is based on increasing average max flow to all the destinations. This ensures that sources which have lesser resources get priority for calculation of trees as the algorithm inherently favors sources whose constraints are verified first. The sources which have a higher max flow have more flexibility for choosing their paths. In this stage, sources undergo three steps.

a) Evaluation of multicast flow constraint for disjoint trees

b) Grouping based on the amount of protection offered (n-k)

c) Evaluation of multicast flow constraints for the shared trees

Before we discuss these steps in detail, we refer the reader to Table 3.1 which gives a list of all the symbols that will be used in this section.

3.4.2.1 Evaluation of multicast flow constraint for disjoint trees

Every session requires k disjoint multicast trees from the source to the various destinations. Additionally since the scheme supports breakdown of up to n-k paths from any source at any given time, these k paths have to be disjoint from the multicast trees of any other session. Hence it is imperative to prune the network of edges; belonging to the current session, before proceeding to the next (Line 4, A-3.3). Determination of multicast flow is accomplished using a heuristic method described in the form of an algorithm. From the definition of multicast flow every session needs to

have a multicast flow value of at least k^*r, where r is the number of destinations (Line 3, A-3.3).

Multicast Flow Calculation. This algorithm is used to calculate multicast flow using a heuristic approach. It consists of three steps

1) Generation of flow combination Set

Calculate flow paths from a given source to various destinations. Generate all possible combinations of flow paths by taking only one flow each from every destination's flow path set.

2) Rearranging the set based on throughput savings

Rearrange the flow combination set based on the assumption that the best flow combination is the one which has the least percentage of edges common with any other flow combination,

3) Finding the multicast flow value

Every flow combination contributes to a multicast flow, a value equal to ***r***. Find if the source has enough multicast flow

3.4.2.2 Source Grouping based on the amount of protection offered (n-k)

Sources are divided into groups (U), based on the amount of protection they can afford. The U_j^{th} group offers a protection of n-k-j (Line 25 - 30, A-3.3). Sources which can meet the constraint of $M^F = k^*r$ are grouped into the default set U_0 (Line 5, A-3.3). As mentioned before, protection at reduced cost is made available to sources that cannot meet the previous constraint (Line 7 - 10, A-3.3). The throughput k is reduced for such sources to maintain n-k constant, provided it stays above k^{th}_i which is the minimum throughput threshold for the i^{th} source (Line 8 - 14, A-3.3). If this step helps

maintain the same protection level the source goes into the U_0 group. Otherwise, it is segregated into the U_j^{th} group (where j is as low as possible) depending on the amount of protection it can afford. (Line 15 - 20, A-3.3).

Algorithm 3.3 (A-3.3)

Algorithm Disjoint Tree Source Filtering

This algorithm groups sources based on the amount of protection they offer. Sources unable to meet the constraints are accommodated by reducing throughput. If they still cannot satisfy the conditions a reduced protection is offered to the source. Sources that cannot meet even the last constraint get discarded.

Input Network Graph G

Destination Set T

Source Set $S`$

Output Source Group set U

Network Graph $G`$

```
1   G` ← G
2   foreach i← 1 to s` do
3       if ∃ f'm(Si ,T`) > r* k  , Si ∈S`in the graph G`,
4           G` ← G`– {E(f'm(Si ,T`)) }: f'm(Si ,T`)= r* k
5           U0 ←U0 ∪ Si
6       else
7           ftemp ← f'm(Si ,T`)/r
8           if (ftemp >K^th_i )
9               n i ← ftemp
10              k i ← ftemp - min((ftemp-K^th_i),(n-k))
11          else
12              S`← S`- { Si }
```

```
13            continue
14        endif
15        flag ←0
16        foreach j← 1 to g  do
17            if (n_i - k_i)= (N^gr_j– K^gr_j )
18                U_j←U_j∪ S_i
19                Set flag←1
20                break
21            endif
22        end for
23        f^r_m(S_i ,T`) ← r* k_i
24        G` ← G`– {E(f^r_m(S_i ,T`)) }
25        if flag =0
26            g ← g + 1
27            N^gr_g ← n_i
28            K^gr_g ← k_i
29            U_g← ∅
30            U_g←U_g∪ S_i
31        endif
32    endif
33 end for
```

Algorithm 3.4 (A-3.4)

Algorithm **Multicast Flow Calculation**

This algorithm is used to calculate multicast flow using a heuristic approach. It consists of three steps

1) Generation of flow combination Set

2) Rearranging the set based on throughput savings

3) Finding the multicast flow value

Input Network Graph G

Destination Set T

Source S_i

Output M^F

---

1)

1 $G_{tmp} \leftarrow G$

2 **foreach** $x \leftarrow 1$ to r **do**

3 $m \leftarrow 0$

4 **while** $(\exists f(S_i, t_x) > 0)$

5 **Find** $f(S_i, t_x) = 1 : E(f(S_i, t_x)) \in G_{tmp}$

6 $G_{tmp} \leftarrow G_{tmp} - E(f(S_i, t_x))$

7 $m \leftarrow m+1$

8 $f^m{}_{ij}x \leftarrow f(S_i, t_x)$

9 **end while**

10 $G_{tmp} \leftarrow G$

11 $m_x \leftarrow m$

12 **end for**

Calculate

13 a) $F^{k1\ k2 \ldots\ kr} \leftarrow f^{k1}{}_{ij}1 \cup f^{k2}{}_{ij}2 \ldots \cup f^{kr}{}_{ij}r : 0 < k_1 \leq m_1,\ 0 < k_2 \leq m_2 \ldots 0 < k_r \leq m_r,\ i \in S,$

14 $j^a \in T,\ 0 < a \leq r$

15 b) $O(F) \leftarrow \{ F^{11\ldots1}, F^{11\ldots2} \ldots\ldots F^{m1m2\ldots mr} \}$

16 c) $P(F^{k1\ k2 \ldots\ kr}) \ \forall\ k_1, k_2 \ldots\ k_r, 0 < k_1 \leq m_1,\ 0 < k_2 \leq m_2 \ldots 0 < k_r \leq m_r$

17 d) $C(F^{k1\ k2 \ldots\ kr}) \leftarrow 1 - \dfrac{E(f^{k1}{}_{ij}1 \cup f^{k2}{}_{ij}2 \cup f^{kr}{}_{ij}r)}{E(f^{k1}{}_{ij}1) + E(f^{k2}{}_{ij}2) + E(f^{kr}{}_{ij}r)}$

18

2) **Rearrange the set** $O(F)$:

19 **If** $F^a, F^b \in O(F) : a < b$ **then** $P(F_a) \leq P(F_b)$ **and**

20 **if** $P(F_a) = P(F_b)$ **then** $C(F_a) \leq C(F_b)$

3) **Finding the multicast flow value**

21 $M^F \leftarrow 0,\ i \leftarrow 0$

22 **while**($|O(F)| \neq 0$)

23 $F_i \leftarrow O_i(F)$

24 **If** $F_b \cap F_i \neq \emptyset$, $\forall F_b \in O(F)$, **then**
25 $O(F) \leftarrow O(F) - \{F_b\}$
26 **endif**
27 $M^F \leftarrow M^F + 1$
28 $i \leftarrow i + 1$
29 **end while**

3.4.2.3 Evaluation of flow constraints for the shared trees

We need to find enough flow for the existence of n-k shared trees (Line 5, A-3.5). So we take every source (i) and find if it has at least $r^* (n_i - k_i)$ multicast flow. The way partitioning is done is such that a source belonging to the U_j^{th} group will be part of exactly n-k-j shared trees. Sources belonging to the j^{th} group which do not have at least $r^*(N^{gr}_j - K^{gr}_j)$ multicast flow go through the following steps:

1) Some of their k trees are shifted to the shared set (provided they are still over the k^{th} limit, Line 8-13, A-3.5).

2) If the source does not have enough trees then we try to reduce n-k for the source and shift it to a different group (compromising throughput for protection, (Line 13-30), A-3.5).

3) If the source does not have trees and reducing n-k reduces the throughput below k^{th} threshold, then we discard the source (Line 32-33, A-3.5).

The algorithm for this step is described in 'Shared Tree Constraint based Source filtering'

Algorithm 3.5 (A-3.5)

Algorithm Shared Tree Constraint based Source filtering

This algorithm is used to calculate if the sources which have been segregated into groups have enough flow left to be able to support shared trees

Input Network Graph $G`$

Destination Set $T`$

Source group set U

Output Refined Source group set U

```
1   foreach j← 1 to g do
2       i← 0
3       F_j ← U_j
4       while F_j≠ ∅ do
5           if  ∃ f^r_m(F_j i , T`)≥ r *(n_i - k_i), in the graph G`,
6               F_j ← F_j - F_j i
7           else
8               n_i   ← k_i+ (f^r_m(F_j i , T`)/r)
9               k_i^old ← k_i
10              k_i   ← max((n_i-(N^gr_j – K^gr_j)), K^th_i)
11              F_j  ← F_j - F_j i
12              U_j  ← U_j - F_j i
13              if (n_i - k_i > 0)
14                  flag ←0
15                  foreach d← 1 to g  do
16                      if(n_i - k_i)= (N_d – K_d )
17                          U_d ←U_d ∪ F_j i
18                          Set flag←1
19                          break
20                      endif
21                  endfor
22                  f^r_m(F_j i , T`) ← r* (k_i^old -k_i)
23                  G` ← G`+ {E(f^r_m(F_j i, T`)) }
24                  if flag =0
25                      g ← g + 1
```

```
26                    N^gr_g ← n_i
27                    K^g_g ← k_i
28                    U_g ← ∅
29                    U_g ← U_g ∪ F_j i
30                endif
31            else
32                f^r_m(F_j i, T`) ← r* k_i^old
33                G`← G`+ {E(f^r_m(F_j i , T`)) }
34            endif
35            i←i+1
36        endif
37    endwhile
38 endfor
```

3.5 Steiner Tree approach for Multicast tree generation

Connections have to be provisioned on the disjoint and shared path sessions once the feasibility of the scheme is verified. This is accomplished by establishing multicast trees from every source to the predefined set of destinations. Since multicast trees shared between sources require higher connectivity, the shared tree connections are established prior to the disjoint set of trees.

3.5.1. Shared Tree Generation

The network coded data would be sent on shared multicast trees. Throughput and protection compromise would have yielded source sessions each with a different number of shared trees. Every source hence needs to be a part of a different number of disjoint trees. The same is accomplished using Shared Tree Generation algorithm. It involves two sub steps

1) Algorithm to ensure that every source is part of the required number of shared trees.

2) Algorithm for finding out the multicast tree using the Steiner tree approach.

Algorithm 3.6 (A-3.6)

Algorithm Shared Tree Generation Algorithm

This algorithm is used to establish the shared multicast trees.

Input Network Graph G

Destination Set $T`$

Source group set U

Output Shared trees G_h^{sh}

Network Graph $G`$

Step 1:Establishing Shared trees

1 $G` \leftarrow G$
2 **foreach** $h \leftarrow 0$ *to* q-*1* **do**
3 $y \leftarrow 0$
4 $V \leftarrow T`$
5 **while**(h $< N_y$-K_y)
6 $V \leftarrow \{ V \cup U_y \}$
7 $y \leftarrow y+1$
8 **endwhile**
9 **Find a Steiner tree** G_h^{sh} **for vertices defined by** V **in** $G`$
10 $G` \leftarrow G` - G_h^{sh}$
11 **end for**

Step 2: Finding a Steiner tree

12 **Select** $v` \in V$
13 $V \leftarrow V - \{ v` \}$
14 $G_h^{sh} \leftarrow \{ v` \}$

15 **while**($V \neq \emptyset$) do
16 **Select** $v` \in V$, $g` \in G_h{}^{sh}$: $d(g`, v`) = \min(d(g, v))$, $\forall v \in V$ **and** $\forall g \in G_h{}^{sh}$
17 $G_h{}^{sh} \leftarrow G_h{}^{sh} \cup v`$
18 $V \leftarrow V - \{ v` \}$
19 **end while**

3.5.2. Disjoint Tree Generation

In this step, the k primary disjoint set of trees required by every source are established. The multicast trees required are generated using the Steiner tree approach mentioned in the previous section.

Algorithm 3.7 (A-3.7)

Algorithm Disjoint Tree Generation Algorithm

This algorithm is used to establish the disjoint trees required by every source.

Input Network Graph $G`$

Destination Set $T`$

Source group set U

Output Disjoint trees G_h

1 $S``\leftarrow \emptyset$
2 **foreach** $p \leftarrow 1$ *to* g
3 $S`` \leftarrow S`` \cup U_p$
4 **end for**
5 **foreach** $i \leftarrow 1$ *to* $|S``|$ do
6 $V \leftarrow T` \cup S_i``$
7 **foreach** $h \leftarrow 1$ *to* k_i do
8 **Find a Steiner tree** G_{ih} **for vertices defined by** V **in** $G`$
9 $G` \leftarrow G` - G_{ih}$

10 **end for**
11 **endfor**

3.6 Coefficient Selection for Coding

3.6.1 Vandermonde matrix

Throughput improvements are obtained by network coding data on the shared trees. Network coding is accomplished by selecting coefficients from the Vandermonde matrix. The structure of an $x*y$ Vandermonde matrix is as shown below.

$$\begin{pmatrix} 1 & 1 & .. & .. & .. & ... & 1 \\ \lambda_1^{1} & \lambda_2^{1} & .. & .. & .. & .... & \lambda_x^{1} \\ \lambda_1^{2} & \lambda_2^{2} & .. & .. & .. & .... & \lambda_x^{2} \\ .. & .. & .. & .. & .. & .... & .. \\ .. & .. & .. & .. & .. & .... & .. \\ \lambda_1^{y-1} & \lambda_2^{y-1} & .. & .. & .. & .... & \lambda_x^{y-1} \end{pmatrix}$$

The coefficients (λ_is) are chosen randomly from a finite field (F) of size $v \geq x$. Every shared protection path is assigned a unique λ_i. Data packets belonging to different sources but which will be transmitted on the same shared tree would use the λ_i assigned to it, albeit a different power (of λ_i) for each packet. ie

$\forall \lambda_{i1}^{j1}, \lambda_{i2}^{j2} \in \lambda_i^j$,

i1 = i2, for all packets coded on the same shared tree,

i1 ≠ i2, for packets coded on different shared trees.

j1 ≠ j2, for all packets coded on the same shared tree.

The field size required for coefficient selection is established as follows. There are a total of s sources with k disjoint paths and n-k shared paths to a given destination.

Since there are k unknowns for every source, $k*s$ equations are obtained from the various disjoint paths. But in these equations, the unknowns are sent without any coding on them and hence have a coefficient of 1. Additionally, the shared paths will contribute another n-k equations in the best case (where all the source packets are combined into one packet for every shared path) or worst case $(n-k)*s$ equations. Either way this results in n-k linearly independent equations at the receiver (because the $(n-k)*s$ equations can be always be combined at the receiver to form n-k equations where every equation has a component belonging to every source).Hence the field size required would be n-k+1.We have to note here that for any given shared path, at a common node the packets can be combined in an exclusive manner (The example in the next section describes how this is accomplished). That is network coding at intermediate nodes is a trivial operation.

3.6.2 **Coefficient Assignment Example**

Consider the case of two sources X and Y, each with three primary packets *x1, x2, x3* and *y1,y2* and *y3* (represented by Columns 1-6 below in the coefficient matrix). Let $n-k = 1$. The 7th and the 8th column in the below matrix are indicative of the shared packets which would be sent from X and Y respectively.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \lambda_7^1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \lambda_7^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \lambda_7^3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \lambda_7^4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \lambda_7^5 \end{pmatrix}$$

Note that since the n-k shares generated by sources will be linearly combined at the common node, we need not have a different λ coefficient for column 8. (We can use a different power of λ_7 for the same.)

So if any of the 7 paths breaks down, we will be able to recover the original data using the six linearly independent equations.

The original packet matrix is

[x1 x2 x3 y1 y2 y3]

The coefficient matrix would be

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & \lambda_7^1 \\ 0 & 0 & 1 & 0 & 0 & 0 & \lambda_7^2 \\ 0 & 0 & 0 & 1 & 0 & 0 & \lambda_7^3 \\ 0 & 0 & 0 & 0 & 1 & 0 & \lambda_7^4 \\ 0 & 0 & 0 & 0 & 0 & 1 & \lambda_7^5 \end{pmatrix}$$

3.7 Time Complexity

In this section we discuss the time complexity of the algorithms mentioned previously. The notations which will be used are mentioned below for the readers' reference.

N - Total nodes in the Graph

$|E|$ - Total number of edges in the graph

F_{max} - Maximum flow possible between any two nodes in the network. Since each edge is of unit capacity .the maximum flow can be equal to the number of edges ($|E|$)

Algorithm 3.1

This algorithm runs through all the destinations .Worst case time $O(N)$.

Algorithm 3.2

The Ford Fulkerson Algorithm is used to calculate flow between every source and every receiver ($O(N^2)$). The running time of Ford Fulkerson Algorithm for calculating max flow is $O(\ |E| * F_{max}) = O(\ |E| * |E|)$. Hence we have complexity for this algorithm as $O(N^2 * |E|^2)$.

Algorithm 3.4

For Step 1, calculating max flows between the given source and all receivers will take $O(N * |E|^2)$. Generating all possible flow combinations will take $O((F_{max})^r) = O(|E|^N)$. Calculation of $O(F^a)$ (ordering of flow combinations) including calculation of $P(F^a)$and $C(F^a)$ would be $O(|E|^2 + (|E|^N\ log\ (|E|^N))$. The last step (3) takes $O(|E|^N)$. Total complexity here is $O(N * |E|^2 + (|E|^2 + (|E|^N\ log\ (|E|^N) + |E|^N) =$

$O(N * |E|^2 + (|E|^N\ log\ (|E|^N))$

Algorithm 3.3

Lines 3 -9 will have the same complexity as algorithm 3.4 which is $O(N * |E|^2 + (|E|^N\ log\ (|E|^N))$. Lines 16 – 22 will take $O(N)$. These get repeated $O(N)$ times. The complexity of this algorithm is $O(N*(\ N * |E|^2 + (|E|^N\ log\ (|E|^N)) + N))$

$= O(N^2 * |E|^2 + \ N*(|E|^N\ log\ (|E|^N)))$

Algorithm 3.5

Lines 16-22 will have the same complexity as algorithm 3.3. Line 5 will take a different complexity as the multicast flow is calculated using max flow and hence takes $O(N * |E|^2)$. Total Complexity is $O(N*((N * |E|^2) + N) = O(N^2 * |E|^2)$.

Algorithm 3.6

The Steiner tree calculation will take $O(N^3 * |E|)$. The total complexity is $O(N^4 * |E|)$.

Algorithm 3.7

Taking into account the Steiner Tree calculation and that it runs for every source and every disjoint path, $O(N^5 * |E|)$ is the complexity.

The total complexity of verifying all the constraints is given by

$O(N).+ O(N^2 * |E|^2) + O(N^2 * |E|^2 + N^*(|E|^N \ log \ (|E|^N))) + O(N^2 * |E|^2)$

$= O(N^2 * |E|^2 + N^*(|E|^N \ log \ (|E|^N)))$

The total complexity of building all the required paths is given by $O(N^4 * |E|) + O(N^5 * |E|) = O(N^5 * |E|)$

3.8 Summary

In this chapter we discussed the problem in detail and established algorithms for the verification of flow constraints as well as for generating the multicast trees. We also looked at the coefficient assignment for network coding and the time complexity of the various algorithms.

CHAPTER 4. SIMULATION RESULTS

4.1.Introduction

We compare our scheme against the original *(k,n)* scheme of erasure coding . The proposed approach works better as demonstrated by results. We look at throughput, cost of provisioning multicasts sessions and the number of sources that the proposed approach can support in different network conditions.

4.2 Cost of Provisioning Sessions

Since the number of disjoint trees has decreased because of sharing trees across source sessions, substantial reduction is seen in the cost for provisioning multicast sessions. We consider a scenario where the network connectivity is sufficiently high. To compare with erasure coding fairly, neither throughput compromise nor protection compromise is employed in this example. The number of nodes in the network chosen is 45 ($N = 45$) with an average Nodal Degree equal to 24. The experiment was repeated for a set of 50 random networks with the above characteristics. The total number of paths that had to be supported was six (n=6) and the number of disjoint paths was four (k=4). The multicast session provisioning costs are compared with erasure coding by varying the number of sources and destinations that need to be supported. The comparison is shown in the table 4.1 and also in Figure 4.1. The plot includes the mean as well as the confidence intervals for confidence levels of 95. This confidence intervals is also shown in Table 4.2 (a). The Figure 4.2 shows the savings that are obtained in terms of cost by the proposed scheme over conventional erasure coding. It can be observed that as the number of sources and destinations in the network increase, the savings also increase. For s=4, the savings for the proposed

scheme increases from 5% all the way up to 20% as destinations are added. This is understandable since the difference between the number of edges in the shared trees and the erasure coding's extra disjoint trees will keep increasing with increase in the number of nodes that need to be supported. The hyphen (-) in the various tables indicate that the corresponding scheme could not support the particular configuration of sources and destinations.

	Cost of Provisioning											
Dest(*r*)	**1**		**2**		**3**		**4**		**5**		**6**	
Sources(*s*)	**EC**	**EC+NC**	**EC**	**EC+NC**	**EC**	**EC+NC**	**EC**	**EC+NC**	**EC**	**EC+NC**	**EC**	**EC+NC**
2	14.8	14.48	26.46	24.3	36.66	33.24	45.32	40.94	54.24	48.54	62.48	55.5
3	22.24	21.34	39.96	35.7	55.62	48.4	69.62	60.28	83.46	71.26	96.54	81.74
4	29.84	28.28	53.64	46.98	75.34	64.08	94.32	79.84	113.52	94.44	130.98	108.34
5	37.68	35.26	67.96	58.8	95.84	80.16	-	99.58	-	117.82	-	135.46
6	-	42.64	-	70.62	-	96.18	-	-	-	-	-	-

Table 4.1: Cost of Provisioning multicast sessions for Erasure Coding (EC) and the proposed scheme - a combination of Erasure and Network Coding(EC+NC).

	Cost of Provisioning with 95% CL											
Dest(*r*)	**1**		**2**		**3**		**4**		**5**		**6**	
Sources(*s*)	Mean	CI	Mean	CI	Mean	CI	Mean	CI	Mean	CI	Mean	CI
2	14.480	0.138	24.300	0.419	33.240	0.501	40.940	0.536	48.540	0.579	55.500	0.600
3	21.340	0.131	35.700	0.478	48.400	0.589	60.280	0.620	71.260	0.548	81.740	0.737
4	28.280	0.136	46.980	0.606	64.080	0.632	79.840	0.643	94.440	0.699	108.340	0.874
5	35.260	0.134	58.800	0.703	80.160	0.736	99.580	0.760	117.820	0.812	135.460	1.030
6	42.640	0.281	70.620	0.786	96.180	0.976	-	-	-	-	-	-

Table 4.2 Cost of Provisioning multicast session using proposed scheme with confidence level of 95%

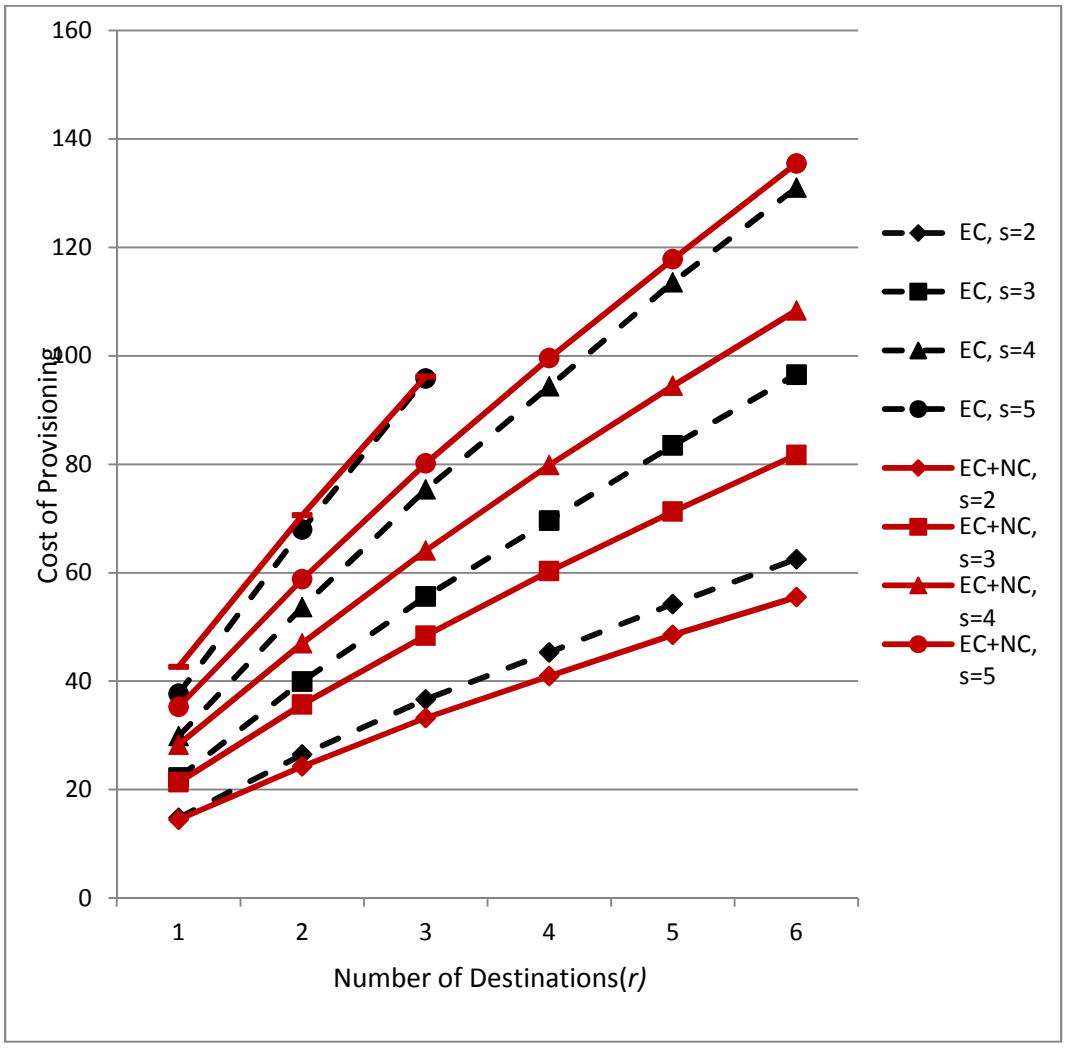


Figure 4.1 Comparing cost of provisioning multicast session for Erasure Coding (EC) and proposed approach- a combination of Network and Erasure Coding

4.3. Throughput

The proposed approach allows for compromise in throughput and protection. If the number of sources is large and cannot be supported at the given protection and throughput, the scheme allows us to take the following directions.

1) Reduce throughput in order to accommodate the source (Tc),

2) Reduce the amount of protection offered to the source (Pc), or

3) Reduce throughput and protection (Tc+Pc)

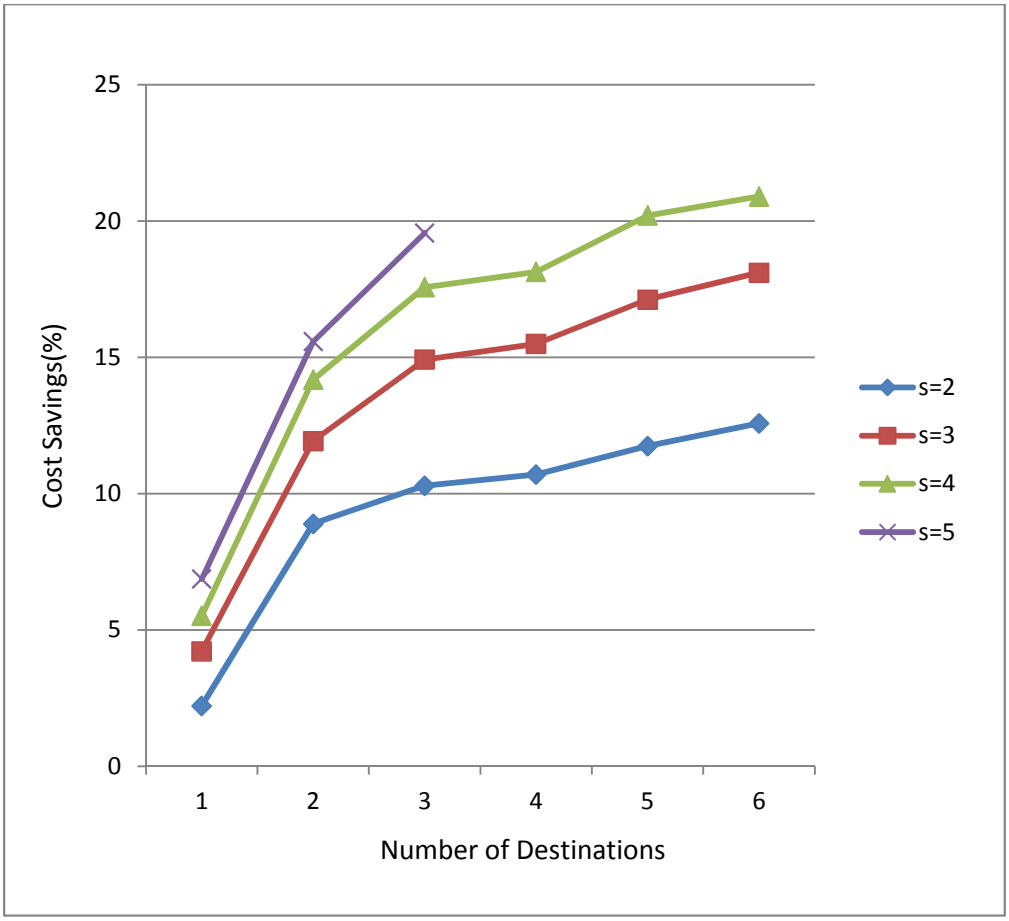


Figure 4.2 Cost Savings of proposed Scheme over erasure coding.

A measure of how good these approaches work is determined using mean throughput obtained while supporting as many sources and destinations as possible. The random networks chosen for this scenario consisted of 40 nodes with an average nodal degree of 20. Two types of simulation setups are considered.

a) In the first case, the variance of throughput with increase in number of destinations was observed. This has four sources and n=5 and k=3.

b) In the second case, the number of sources was changed. There are four destinations to which data has to be multicasted and n=6, k=4.

For case (a), the mean throughput for the proposed scheme is better than the original erasure coding scheme. The increase in the number of destinations does not affect the throughput too much. Our approach involving throughput and protection compromise performs better than all the other approaches. Table 4.3 shows the performance of the various approaches .The same is plotted in Figure 4.3.

	Mean throughput(Data Units)			
Num of Destinations	**EC**	**EC+NC**	**EC+NC+Tc**	**EC+NC+Tc+Pc**
5	2.19	2.955	2.955	2.955
6	2.13	2.865	2.885	2.885
7	2.055	2.535	2.565	2.645
8	1.815	2.535	2.565	2.66
9	1.77	2.175	2.265	2.425
10	1.71	2.205	2.26	2.41

Table 4.3 Comparison of Mean throughput/source with variable destinations for

a) Erasure Coding (EC)

b) Erasure Coding + Network Coding (EC+NC)

c) Erasure Coding+ Network Coding+ Throughput Compromise (EC+NC+ Tc)

d) Erasure Coding+ Network Coding+ Throughput Compromise+ Protection Compromise (EC+NC+Tc+Pc)

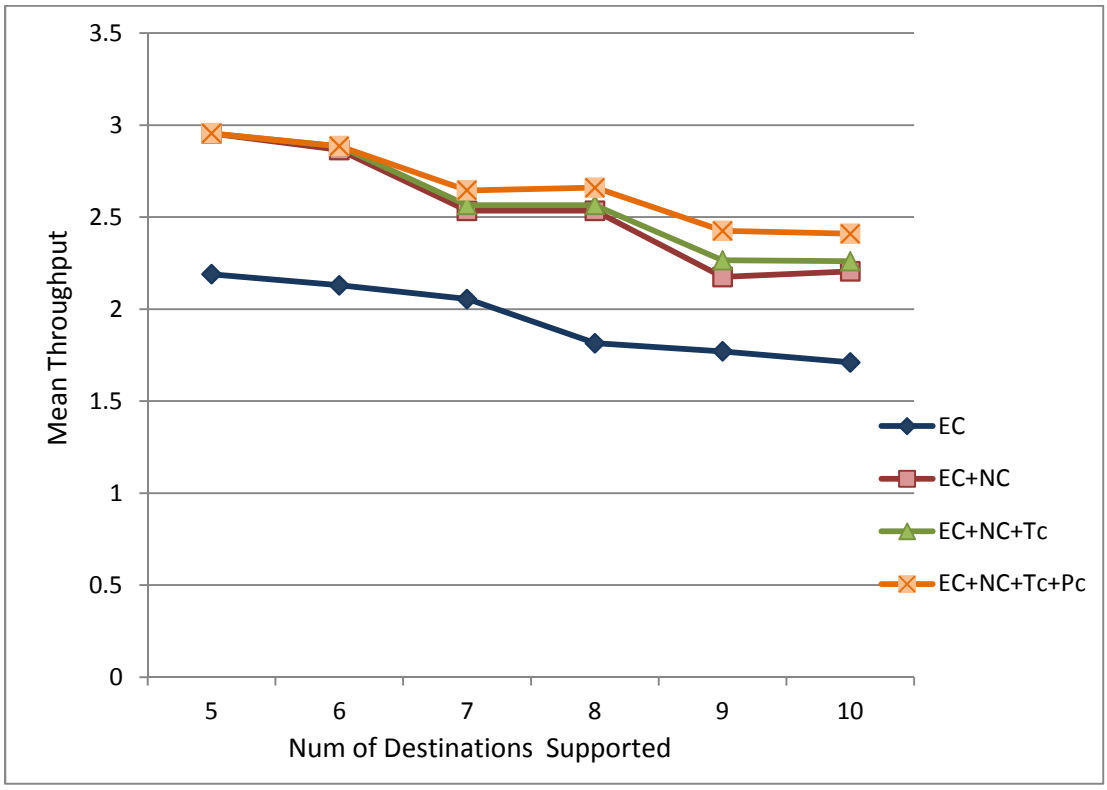


Figure 4.3 Plot of Mean throughput/source with variable number of destinations for EC, EC+NC, EC+NC+Tc, EC+NC+Tc+Pc

For case (b), there is a substantial drop in throughput as the number of sources increase , for all the approaches. When EC and EC+NC cannot support a source then the throughput anyway becomes less as the number of sources increases. For the approaches which involve decreasing throughput to support more sources, the mean throughput is bound to be less. The approach which performs best here is the one where only protection is compromised to support more sources. EC+NC+Pc does better than EC and EC+NC because it supports more sources than both of these approaches. It does better than EC+NC+Tc and EC+NC+PC+Tc because it does

not compromise throughput to support more sources which either approaches do. As is evident from the Table 4.4 and Figure 4.4, EC+NC+Pc always does better than the other approaches.

	Mean Throughput(Data Units)				
Num of Sources	EC	EC+NC	EC+NC+Tc	EC+NC+Tc+Pc	EC+NC+Pc
3	2.853	3.867	3.867	3.867	3.947
4	2.260	2.100	2.370	2.830	3.040
5	1.696	1.568	1.824	2.296	2.416
6	1.440	1.147	1.347	1.837	1.973
7	1.223	1.051	1.274	1.651	1.703
8	1.080	1.140	1.380	1.503	1.490
9	0.987	1.120	1.260	1.331	1.316
10	0.888	1.016	1.138	1.192	1.192
11	0.785	0.960	1.058	1.091	1.091
12	0.670	0.873	0.988	1.002	0.987

Table 4.4 Comparison of Mean throughput/source with variable number of sources for EC, EC+NC, EC+ NC+ Tc, EC+NC+Tc+Pc and EC+NC+Pc

4.4. Sources supported for multicasting

While the basic scheme of EC+NC attempts to decrease the cost of provisioning the multicast sessions, other approaches like EC+NC+Tc try to maximize the number

of sources that could be supported. Essentially they try supporting those sources that cannot meet the required constraints of EC+NC. Since throughput compromise is something that can be implemented for the original erasure coding too, it will be interesting to see how the two algorithms (EC and EC+NC) fare w.r.t supporting maximum number of sources given both are allowed to compromise their throughput

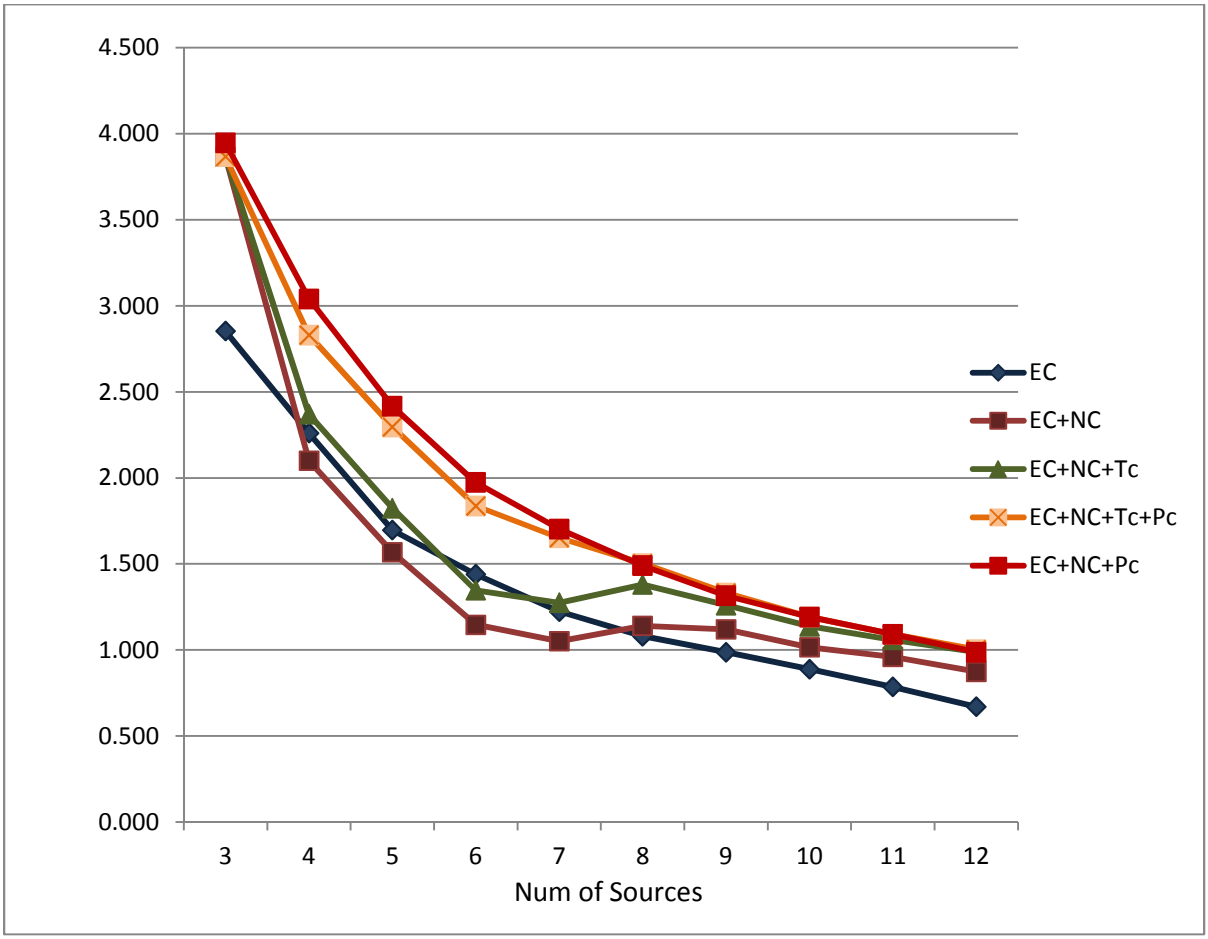


Figure 4.4 Plot of Mean throughput/source with variable number of sources for EC, EC+NC, EC+ NC+ Tc, EC+NC+Tc+Pc and EC+NC+Pc

. The number of nodes considered in the random network is 40 (N=40). The average nodal degree is 22.The results were taken over 50 iterations. The number of specified

destinations is 3 (r=3). If the number of destinations is allowed to be reduced, then both the approaches would be able to support more sources. Hence the comparison is given for both cases, with and without destination reduction.

a) The focus is on supporting as many sources as possible in spite of reduction in supported destinations.

b) Max number of sources that can be supported such that all the destinations are still part of the scheme.

For case (a), EC+NC+Tc is able to support around 10 sources when the protection n-k is 1, whereas EC+Tc is able to support only 8 sources. As the protection increases, the EC+NC+Tc supports many more sources than EC+Tc. For a protection of n-k =3, EC+NC+Tc supports 8.5 sources on the average whereas EC+Tc is able to support only 3.6 sources, i.e., EC+NC+Tc is able to support more than 135% of what EC+Pc can support. The difference between the number of sources supported by EC+NC+Pc and EC+Tc almost remains constant following this. Figure 4.5 (a) shows the plot and Table 4.5 (a) compares the actual values of sources supported .The graph also indicates the confidence interval range values for a confidence level of 95%. (MIN_95_CL, MAX_95_CL)

For case (b), EC+NC+Tc is able to support an average of around 8 sources when the protection n-k is 1 ,whereas EC+Tc is able to support only 6 sources. A fairly constant difference of around 3 sources is maintained between EC+NC+Tc and EC+Tc .This can be seen in the plot shown in Figure 4.5 (b) and the actual values in table 4.6 (b).The confidence interval range value for 95 % confidence level is also plotted on the graph for reference. (MIN_95_CL , MAX_95_CL)

	NUM OF SOURCES SUPPORTED					
	EC+NC+Tc			EC+Tc		
Protection(n-k)	Mean	MIN_95_CL	MAX_95_CL	Mean	MIN_95_CL	MAX_95_CL
1	9.86	9.74899	9.97101	7.9	7.81684	7.98316
2	9.44	8.94989	9.93011	5.8	5.68913	5.91087
3	8.46	8.19292	8.72708	3.6	3.46421	3.73579
4	8.1	7.46975	8.73026	3.8	3.68913	3.91087
5	6.62	5.96666	7.27334	2.54	2.40185	2.67815
6	6.94	6.29252	7.58748	2.8	2.68913	2.91087

Table 4.5 (a) Comparison of Number of sources supported with flexible number of destinations for EC+NC+Tc and EC+Tc

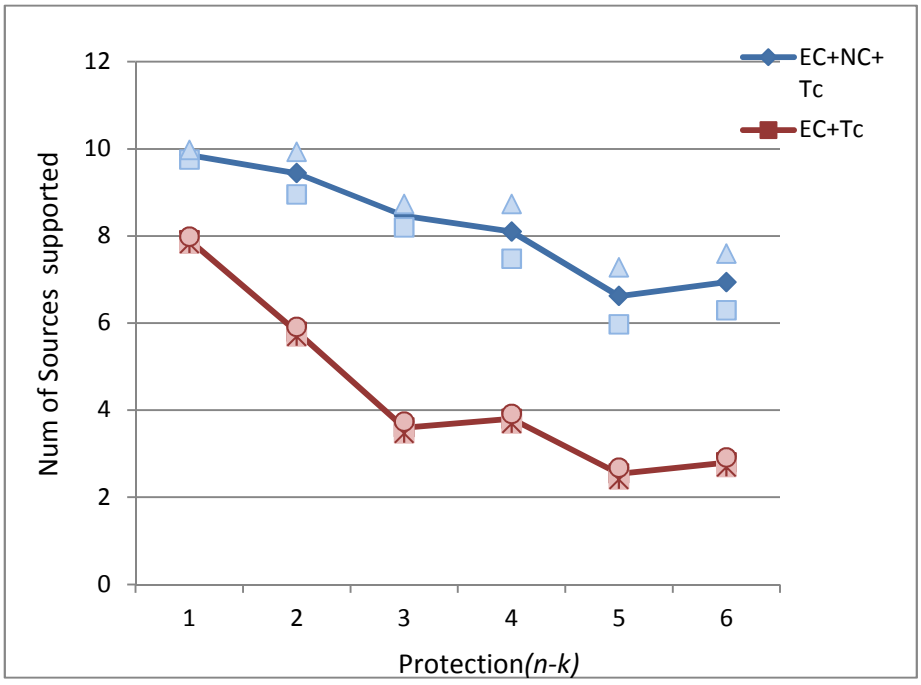


Figure 4.5 (a) Plot comparing number of sources supported with flexible number of destinations for EC+NC+Tc and EC+Tc

	NUM OF SOURCES SUPPORTED					
	EC+NC+Tc			EC+Tc		
Protection(n-k)	Mean	MIN_95_CL	MAX_95_CL	Mean	MIN_95_CL	MAX_95_CL
1	8.1	7.98571	8.21429	5.98	5.94119	6.01881
2	7.54	7.19732	7.88268	4.68	4.5507	4.8093
3	6.84	6.58986	7.09014	3.6	3.46421	3.73579
4	5.66	5.02644	6.29356	2.94	2.87417	3.00583
5	5.48	5.11127	5.84873	2	2	2

Table 4.5 (b) Comparison of Number of sources supported with fixed number of destinations for EC+NC+Tc and EC+Tc

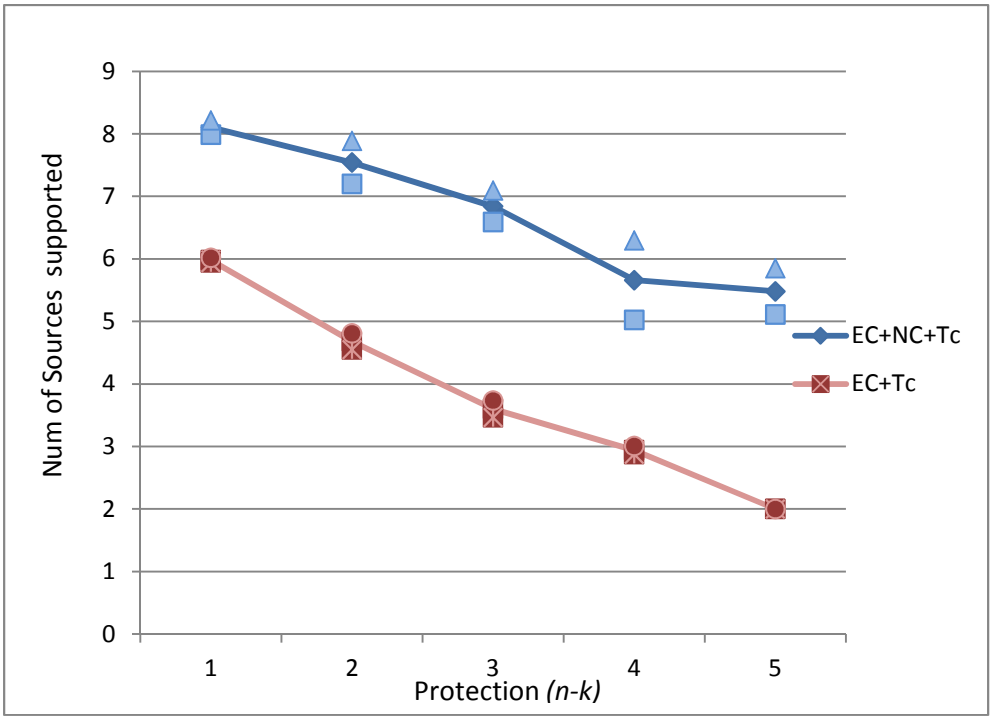


Figure 4.5 (b) Plot comparing number of sources supported with fixed number of destinations for EC+NC+Tc and EC+Tc

4.5 Summary

In this chapter we looked at the performance of the proposed approach against conventional erasure coding .Specifically we compared the results for savings in cost, throughput and number of sources that could be supported by the two approaches. We saw that the proposed approach along with the enhancements of throughput and protection compromise does reasonably well job of providing protection at a reduced cost. We saw throughput savings of up to 20% compared to erasure coding. We also saw that the proposed scheme was able to support more than 100% of the sources supported by erasure coding.

CHAPTER 5. CONCLUSIONS AND FUTURE RESEARCH

5.1 Proposed Approach Discussion

We have proposed an approach that enables packets from multiple sources to be network coded and sent to multicast destinations. It uses the principle of erasure coding where k out of copies are required at the receiver to be able to decode the data The scheme supports breakdown of any n-k paths of a source at any given time. The recovery is done by utilizing the packets obtained on the primary and shared paths of other sessions. This necessitates that the primary paths of all sources are disjoint. The advantages using our implementation can be briefly described as follows.

1) Throughput savings obtained by network coding packet belonging to different sources. Simulations show an average of 20% savings in bandwidth (provisioning costs).
2) Ability to support sources that cannot meet the stricter flow constraints of the basic scheme, by way of reduced throughput or protection. Hence certain sources can take part in the multicast sessions in spite of having lower connectivity.
3) The network coding at the intermediate nodes is simple xoring and can be accomplished using a Field of size (2). Additionally Field size for generating coded packets at the source is small.
4) The recovery of source packets at the receiver can be instantaneous providing proactive protection to the source sessions.

The drawbacks of the scheme include

1) The proposed scheme imposes a high connectivity requirement on the network. Every source needs to have k disjoint paths to the destination set .Also there needs to be n-k shared trees between the various sources and destination set.
2) Cannot handle sources with different destination requirements as it will result in pollution. (Receivers will receive data from sources they don't need data from.)

5.2 Future Work

The future work that could be having sessions that can have different number of destinations for every source. That is destinations can tolerate a certain amount of pollution as long as their guaranteed the required protection. Also, in this scheme we consider cases where n and k are different from what is specified (Throughput and Protection compromise). Instead of making it a special case, extensions can be made to the existing scheme to be able to support any value of n and k provided the source meets the necessary constraints.

BIBLIOGRAPHY

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung: Network information flow, IEEE Transactions on Information Theory, Vol. 46, pp. 1204–1216, July 2000

[2] S. Alouneh, A. Agarwal, A. En-Nouaary: A Novel Approach for Fault Tolerance in MPLS Networks, Innovations in Information Technology, Nov 2006

[3] A. Fei, J.-H. Cui, M. Gerla, and D. Cavendish: A Dual-Tree Scheme for Fault-Tolerant Multicast, In Proceedings of IEEE ICC'01, Helsinki, Finland, June 2001

[4] J. Cui, M. Faloutsos, M. Gerla: An architecture for scalable, efficient, and fast fault-tolerant multicast provisioning, IEEE Network Magazine, Vol.18, Issue 2, Mar-Apr 2004

[5] R. Koetter and M. Medard: An algebraic approach to network coding, IEEE Transactions on Networking, 2003

[6] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, and L. Tolhuizen: Polynomial time algorithms for multicast network code construction, IEEE Transactions on Information Theory. Vol. 51, No.6, 2005

[7] T. Ho, M. Medard and R. Koetter: An Information-Theoretic View of Network Management. IEEE Transactions on Information Theory, April 2005

[8] S. Aly, A.E Kamal: Network Coding-Based Protection Strategy Against Node Failures, IEEE International Conference on Communications, 2009

[9] Y. Wu: On Constructive Multi-Source Network Coding, IEEE International Symposium on Information Theory, pp. 1349-1353, July 2006

[10] P.Bao-xing, Y. Lu-ming, W. Wei-ping, X. Xiao: Linear Network Coding Construction for Multi-Source Multicast Network, First International Workshop on Education Technology and Computer Science, 2009. ETCS '09.

[11] M. Mohandespour, A.E Kamal: 1+N Protection in Polynomial Time: A Heuristic Approach, IEEE Global Telecommunications Conference, 2010

[12] A. E. Kamal, A. Ramamoorthy, L. Long, S. Li: Overlay Protection Against Link Failures Using Network Coding, IEEE/ACM Transactions on Networking, Issue: 99, 2010

[13] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J.Shi and B. Leong: A random linear network coding approach to multicast, IEEE Transactions on Information Theory, Vol. 52,No. 10, pp. 4413- 4430, 2006

[14] D. S. Lun, M. Medard and M. Effros: On coding for reliable communication over packet networks, in Proceedings of 42nd Annual Allerton Conference on Communication, Control, and Computing, Sept.–Oct. 2004

[15] S.-Y. R. Li, R. W. Yeung, and N. Cai: Linear Network Coding. IEEE Transactions on Information Theory, Vol. 49, pp.:371 – 381, 2003

[16] A.E. Kamal: 1+N Network Protection for Mesh Networks: Network coding-Based Protection using p-Cycles. IEEE/ACM Transactions on Networking, 18(1): 67–80, Feb. 2010

[17] AE Kamal: 1+ N protection against multiple faults in mesh networks. In Proceedings of the IEEE International Conference on Communications , 2007

[18] C. Fragouli, J.-Y. LeBoudec, and J. Widmer: Network coding: An instant primer, ACM Computer Communication Review, vol. 36, pp. 63–68, Jan. 2006

[19] H. Weili, G. Hongyan: A Fault-Tolerant Strategy for Multicasting in MPLS Networks, International Conference on Computer Engineering and Technology, 2009

[20] B. Yang,P. Mohapatra: Edge router multicasting with MPLS traffic engineering, 10th IEEE International Conference on Networks, 2002